NASA CR-134455

NASA-CR-134455) NASIS DATA BASE MANAGEMENT SYSTEM: IBM 360 TSS IMPLEMENTATION. VOLUME 1: INSTALLATION STANDARDS (Neoterics, Inc., Cleveland, Ohio.) 24 23 p HC $3.25 CSCL 09B G3/08

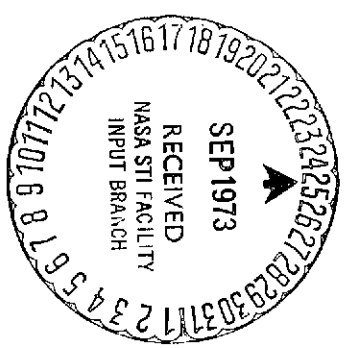
N73-30139

Unclas 13477

NASIS DATA BASE MANAGEMENT SYSTEM - IBM 360 TSS IMPLEMENTATION

I - INSTALLATION STANDARDS

NEOTERICS, INC.



prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NASA Lewis Research Center
Contract NAS 3-14979

TABLE OF CONTENTS

A. INTRODUCTION

The NASIS Development Workbook contains all the required system documentation. The major sections are:

I. Installation Standards

II. System Overviews

III. Data Set Specifications

IV. Program Design Specifications

V. Retrieval System Reference Manual

VI. System Messages

VII. Operating Specifications

VIII. Data Base Administrator Reference Manual

B. INSTALLATION STANDARDS

This section describes the standard approach to preparing system documentation and outlines program design and coding rules and conventions. Included are instructions for preparing all major specifications and suggestions for improving the quality and efficiency of the programming task.

C. SYSTEM OVERVIEWS

The system overviews are a management tool. They are directed toward informing management of a system's capabilities and requirements and are written containing a minimum of technical terminology.

The intent of the overviews is to introduce the system features to interested individuals. Therefore, each major component of the system is represented in the overviews section. Each overview contains a description of the component's activities and role in the overall system. This description includes explanations of new terms and concepts, clarifying charts and diagrams and a discussion of the performance requirements and growth potential of the module. Performance requirements consist of assumptions about the environment (hardware and software) and input-output format.

D. DATA SET SPECIFICATIONS

The data set specifications describe, the content, format, and medium of communication of every data set required by the system. Thus, all relevant information pertinent to a particular data set is prepared in a standard form and centralized in a single document. The standard format is the following:

1. Introduction

   This narrative notes identifying characteristics of the data and its file. It includes layout sheets and information in the following format:

   A. DATA SET NAME: Name as used in the Design Specifications.

   B. CREATED BY: Name of the module in which the data set is originally prepared.

   C. TYPE OF FILE: One of the six identified in part 2 of the data set specifications.

   D. ORGANIZATION: VISAM, tape sequential, etc.

   E. KEY IDENTIFIER (CONTROL FIELD): Name of the field used for access to a record; also, length of key field.

   F. RECORD LENGTH: Byte length; if variable, state the minimum record length.

   G. BLOCKING FACTOR: If unblocked, state '1'.

   H. PURPOSE: Describe the purpose of the data set, its general contents and any peculiarities or special features.

2. Forms

   Because different types of data sets do not require the same descriptive information, several different forms are needed to adequately specify data sets. Each of the six types of data sets is discussed separately below with the specific forms requirements for each indicated.

   a. Punched Cards

      Use the Multiple-Card Layout Form (X24-6599) to designate each separate distinct card format which has fixed fields. Each field should be indicated by vertical lines

surrounding the field. To identify the field, consecutively number each field, starting at the left (e.g., 1, 2,). On a separate page, itemize each field by number and identify the field name and the contents of the field. Define the validation rules for those fields which appear only on the card and do not carry on into the system.

For cards which are free format or contain variable fields, define the card layout using a tabular form which specifies the fieldname, the starting column (if appropriate), the minimum and maximum field lengths, and the content. Also, indicate any field delimiters.

b. Formatted Print-outs

Define the format for each distinct data set using the standard IBM printer spacing chart, Form X20-1776. Print all headings in the exact postions in which they occur. Include any numeric editing or alphabetic character insertions which ALWAYS appear on the report. Should one data set have multiple print formats, use a separate form for each. In any case, each different and distinct LINE format must be indicated at least once for each data set. Be sure to indicate any special end-of-program or end-of-routine print-outs which can occur. In all cases, the layout format shall represent a sample report.

c. Terminal Communication

Most terminal communication is in the form of prompting messages and responses which are fully described in the appropriate Design Specification. Thus, the particular format utilized need not be referenced through a Data Set Specification.

Display outputs in restricted format, however, require description on the Proportional Record Layout Form (X20-1702). Document each distinctly identifiable display format on a separate form. Since terminal displays may not truly be data sets, the introductory page may not contain all the requested information.

Prepare 2 display formats in the manner specified above for print-out formats. Print headers on the form with all fields indicated by X's. Indicate editing marks or constants in their respective positions. Indicate continuous streams of information broken into multiple lines with a block outlined by X's and a printed notation within the block which defines its parameters and contents. Also, mark fields which always appear separate and distinct with a numerical reference. Define each of these fields on a separate page, by number.

d. Tables

Some tables are used to communicate between modules; others are used internally. Most of the internally-used tables should be defined within the appropriate module Design Specification.

Use this section to describe any tables, lists, or structures which are parameters to a routine or pass information between modules. Because of the nature of these elements, no particular form need be used to define them. However, the introductory page shall still be prepared.

Structures can best be described by showing the elements of the structure in an indented hierarchical format, such as the structure would appear in a PL/I program. Define each element of the structure by either a line comment on the form or a narrative description on a separate page.

Lists (arrays) typically contain recurring elemental items, each of the same format and size. To define a list, simply give the dimensions of the list (number of items, state maximum, if variable) and describe the detail for one element. This description includes name, length, and content of each field in a list element.

In many cases, a structure can also be defined in tabular form. A set of related, but unlike, items might be grouped together to form a table. In these instances, the means of description is a columnar document. To standardize the format, apply the

following concept to each element:

1. Indicate the starting character position, assuming the entire set of elements is a continuous character string.

2. Denote the field name.

3. Give the field length.

4. Describe the field content, indicating any particular values assignable to the field.

To segregate the columns, use vertical lines and headers for each column.

e. Non-Data Base Files

The documentation for each data set includes the introductory page and one or more layout forms. If the file contains any fixed-length records, define the specific format on the System/360 Record Layout Worksheet (X20-1711). Indicate each field by vertical lines placed in the appropriate positions on the form. Number the fields consecutively (1, 2, etc.) and, on a separate page, itemize each field by number, with name, length, type (binary, EBCDIC, etc.), and state content beside each field.

Since variable-length records preclude the use of a fixed-format layout form, describe any files containing such records in a tabular format. In using a columnar format, all delimiters or field-length indicators must be mentioned for each applicable field. In some cases, where only a few variable-length fields occur at the end of a record, it may be more beneficial to define the record using both a layout form, for fixed-length fields, and a table, for the variable-length fields.

Some of the files may contain header and/or trailer records. For these include a separate layout form or table to describe the special record formats. In addition, mention any special conditions or considerations (special handling, checkpoints, etc.).

f. Data Base Files:

Define each data set in terms of the set of descriptors for that file. To accomplish the definition, a special descriptor layout has been prepared and shall be used. The first specification in this section is the descriptor format for the descriptors, themselves. Included is the format of the descriptor header.

Detailed information for using the form for the descriptors can be obtained from the descriptor editor user's guide.

E. PROGRAM DESIGN SPECIFICATIONS

1. Overview

This section contains the design specifications for the programs and modules within NASIS. Its purpose is to standardize the preparation of the specifications and to guide the program design.

Each major functional module within the system is a separate entity for documentation purposes. The design specifications shall contain a description of, and specifications for, all detail processing which occurs in the module. Sub-modules, reference tables and data sets which are common to several modules are documented separately.

All modules do not require narrative entries under each heading and for some generalized routines a variance from the standard approach may be necessary. However, in all cases, the unused headings shall still be included in the specifications and a 'NOT APPLICABLE' reference noted. This action removes all doubt that anyone omitted considerations relative to a heading and yields a positive response to each item.

2. Format

The standard outline is shown in the following chart:

A. MODULE NAME

B. ANALYST

C. MODULE FUNCTION

D. DATA REQUIREMENTS

1. I/O Block Diagram

2. Input Data Sets
   a. Parameter Cards
   b. Punched Card Input Files
   c. Input Files
   d. On-Line Terminal Entries

3. Output Data Sets
   a. Output Files
   b. On-line Terminal Displays
   c. Formatted Print-outs
   d. Punched Card Output Files

4. Reference Tables

E. PROCESSING REQUIREMENTS

1. Top Level Flowchart

2. Narrative

F. CODING SPECIFICATIONS

1. Source Language

2. Suggestions and Techniques

The following paragraphs describe the content of each section:

MODULE NAME Heading

State a functional name for the module, e.g., MAINTENANCE. Following the functional name, state a program name, which is eight or fewer characters with the first character an "R", e.g., RDBMNTN. After the program name state the module name, usually the same as the program name minus the leading "R".

ANALYST Heading

Record the name(s) of the individual(s) who prepared the design specifications.

MODULE FUNCTION Heading

Use this heading to give a general description of what the module accomplishes. Limit the narrative to approximately one-fourth page of single-spaced

typed copy.

DATA REQUIREMENTS Heading

Define all the input and output data sets, including any intermediate work-data sets. Also, specify any reference tables used by the module. If the nature of the module is such that some data sets are unknown until execution time, outline the classes of data sets that are expected.

In specifying data sets, a name reference to the proper data set will suffice since all data sets are fully described in their own section of the Workbook. However, while details of the formats of common data sets are not necessary, the processing of the data must still be specified.

I/O BLOCK DIAGRAM Heading

To place the module in the proper perspective and to give an overall picture of the data flow, draw a separate block diagram of the module and all data sets referenced by the module. The symbols for the block diagram shall be obtained from the latest revision of the IBM template (Form X20-8020).

INPUT DATA SETS Heading

Specify all input files. Describe in detail those Job Control cards which may affect the module.

PARAMETER CARDS Heading

Explain the functions which are controlled by each parameter that appears on each parameter card used by the module. In addition, include a table showing the card format and a description of each field's content. If applicable, as for a fixed-position format, standard Multiple-Card Layout Forms may be used. A sample format layout follows:

1. Card Name - DATE

2. Field - 1
   a. Columns 1-2
   b. Contents - Year

3. Field - 2
   a. Columns 3-4
   b. Contents - Month

4. etc.

PUNCHED CARD INPUT FILES Heading

Identify all punched card files defined in the Data Set Specifications section of the Workbook using the appropriate standard file name.

INPUT FILES Heading

Identify all input data sets by the appropriate standard file name.

In the case of generalized modules, data sets cannot be identified by name. However, it is probable that a particular class of data sets is accessed by the module. Write a short paragraph defining the class or classes of input data sets, along with a general indication of how each class might be referenced or manipulated.

ON-LINE TERMINAL ENTRIES Heading

In many modules, certain responses or commands are issued from terminals. Although a particular sequence of fields of data may be expected, the actual format could be free. For purposes of documentation, state all response formats in terms of the generalized free format construction and indicate any necessary field delimiters. Use a table, such as the type specified for parameter cards, for the various entry formats. Since many responses are subject to particular execution time situations, actual values for the responses are specified as part of the Processing Requirements section.

OUTPUT DATA SETS Heading

Output from a module may consist of many different types of information in many different formats. Specify all data sets including terminal displays and card and printer output. Describe in detail those Job Control cards which can affect the module.

OUTPUT FILES Heading

All data base output files, with the possible exception of intermediate work files, are defined in the Data Set Specifications section of the Workbook. Identify by the appropriate standard file name all data sets output by the module.

Where work files are needed, determine whether the intermediate file is utilized by other modules in the system. If so, insert the specifications for the file in the Data Set Specifications section of the Workbook. If not, define the work file in detail, including the following information where applicable:

1. File attributes
   a. organization
   b. recording medium
   c. name

2. Record attributes
   a. key identifier
   b. length
   c. mode (fixed or variable)
   d. blocking factor

3. Field attributes (per each field)
   a. length
   b. mode
   c. data type (alphabetic, etc.)
   d. position in record

4. Other identifying information

In the case of generalized modules, data sets cannot be identified by name. However, it is probable that a particular class of data sets is output by the module. Write a short paragraph to define the class, or classes, of output data sets, along with a general indication of how each class is prepared.

ON-LINE TERMINAL DISPLAYS Heading

Describe all specific display formats generated by this module. In those instances where displays are generalized messages whose values are set at execution time, show maximum field length attributes and data type attributes (alphabetic, etc.) and indicate a field name for reference by the Processing Requirements section of the specification.

Where the display is a graph or has some other non-linear peculiarities, write a short paragraph to identify the function and oddities of the display. In all instances, assign an identifying name to each distinct format so that proper reference can be made within the Processing Requirements section.

FORMATTED PRINT-OUTS Heading

All print data sets generated by this module are defined in the Data Set Specifications section of the Wcrkbcok. Identify them by the appropriate standard file name. No reference need be made to the various fields within the files since the Processing Requirements section of the specifications for this module contains the detailed processing steps necessary to produce the print outs. Should the print out require a special form or particular line spacing not menticned in the Data Set Specifications, describe those requirements.

PUNCHED CARD OUTPUT FILES Heading

All punched card data sets generated by this module are in the Data Set Specifications section of the Workbcck. Identify them by the appropriate standard file name. No reference need be made to the various fields within the file since the Processing Requirements section of the specificatiens for this module contains detailed processing steps to format the cards. Should a specially-designed card be needed which is not mentioned in the Data Set Specifications, describe it.

REFERENCE TABLES Heading

Define all tables needed by the module. The tables can be either local to this module only or global for use by several modules. In the latter case, define the table in detail in the Data Set Specifications section.

The designer's objective is to identify table usage; however, if the table is local to the module, detail table specifications must be stated. In providing this design-level information, the designer shall:

a. identify the table with a suitable name.

b. identify the table organization, specifying calling sequence or cther means of access to the table.

c. define each field in the table by length, mode (F or V) and data type.

d. give content of each of the fields if the

table is used merely for reference. If the table is to be used for internal storage, then specify processing steps in the Processing Requirements section.

If necessary for clarity, use a chart such as Table 1 in the Appendix.

If the table is a true matrix or array of like elements rather than a centralized source of related information, specify the dimensions of the matrix, the format and length of each element, and the number of elements in each dimension. For all tables, precede the table specifications with a brief discussion of its function.

PROCESSING REQUIREMENTS Heading

Use this heading to present the functional specifications of the program. These specifications indicate the functions that the module will have to perform.

The minimum documentation is a top-level flowchart and a module narrative which presents the functional specifications of the module in generally the same order in which they are executed. Use figures, decision tables, and additional lower-level flowcharts for clarity.

TOP-LEVEL FLOWCHART Heading

The flowchart drawn under this heading depicts on an overall basis, the flow of data through the various processing steps within the module. The symbols for all flowcharts drawn in the Processing Requirements section shall be obtained from the latest revision of the IBM template (Form X20-8020).

In designing the flowchart, observe these ground rules:

a. The logical flow of the module proceeds from the upper left corner of the sheet to the lower right corner. If practical, avoid right-to-left and lower-to-upper processing.

b. Use the standard symbols and conform to standard flowcharting conventions.

c. Show the logic of the module as one mainline of processing which governs performance of specific subroutines. Frame each subroutine or sub-function reference in a PREDEFINED PROCESS symbol with the name of the routine.

NARRATIVE Heading

The most critical section of any design specification is that which defines the actual manipulation of the data. For this reason, the narrative shall be written in a manner which is simple and understandable, yet complete in sufficient detail.

The data flow depicted in the top-level flowchart serves as a guide to the sequence of the written specifications. The mainline of the module is defined first, thus presenting a written overview of the processing. Present each of the sub-functions or subroutines, as indicated on the top-level flowchart, under sub headings which reference the name stated on the flowchart.

Within each of the areas, prepare specifications of required processing for that area.

On occasion, complete written description is extremely difficult to prepare. In those cases, supplementary material may be helpful. Decision tables, illustrative charts, tables, and, particularly, detail-level flowcharts may be useful. Any such detail flowcharts are to follow the conventions described in the paragraphs 3.d-f. below. However, since the sole purpose of the Narrative section is to adequately specify ALL the processing required by the module, the written documentation is a primary requirement, while other means of presentation serve merely to clarify the narrative.

Since the Data Requirements Section makes no attempt to define the processing required to use or format I/O, this section is to specify all manipulation of data fields. State formatting of output records, generation of table entries, and manipulation of input records field by field.

CODING SPECIFICATIONS Heading

Although the job of the analyst is to specify what is to be done and not how to do it, certain

aspects of the implementation can often be foreseen. State these points affecting the programming phase.

SOURCE LANGUAGE Heading

The installation may use only one coding language, in which case this heading becomes superfluous. Often, however, one language may be favored over another for a particular module because of some special attributes of the language or efficiency considerations. State the language suggested for usage. If certain sub-modules are programmed in a different language, identify each sub-module and give valid reasons for its difference.

SUGGESTIONS AND TECHNIQUES Heading

While the intent of the Program Design Specifications is not to design the intricate details of a program, it often is the case that the analyst bases various portions of his design on certain implementation techniques. Document any suggestions for efficient methods of coding certain intricate routines and any techniques for handling especially complex processing. Assume the programmer is competent in good coding techniques and, thus, document only special situations.

3. General

In applying the format for writing Program Design Specifications, certain general conventions shall be observed.

a. Type all documentation, including charts, graphs, and tables, but excluding flowcharts, on standard Neoterics, Inc. specifications forms.

b. Insert inapplicable headings and document as "Not Applicable".

c. Write narrative using complete sentences and good paragraph structure.

d. Prepare flowcharts on plain Neoterics paper using the standard symbols on the IBM template number X20-8020. Charts proceed from top to bottom and from left to right.

e. Each page of flowcharts is self-contained.

That is, show a routine from beginning to end cn one page (if possible). Since many routines have too much detail for one page, the first page of the flowcharts for that routine depicts the total flow of the routine with any necessary subroutines indicated as a "predefined process." These subroutines are then charted on other pages. The entire set of flowcharts then is nested where the charts become progressively more detailed. In this method, no interpage connectors are necessary. To provide reference points for any detailed subroutines, each "predefined process" symbol contains the page number of the page showing the detail of the subroutine.

f. Arrowheads shall appear on right to left and bottom to top directional lines. Use of arrowheads cn other lines is optional and should be omitted except where required for clarity.

F. PROGRAMMING CONVENTIONS

1. Overview: This section defines the coding conventions and format for the modules within NASIS. Its purpose is to provide a programming standard which yields code that is consistently structured, adequately documented, and easily readable.

Separate ccnventions are discussed for each language used. Where practicable, however, formats and structures are consistent between the languages.

Prime targets for standardization include comments, statement labels, data area structure, and program names and line numbers. Additional consideration is given to the use of optional symbols and key words for documentation purposes, to placements of the various sections of the coding, and to the identification of program sections or sub-modules via line-spacing and comment headers.

2. Language Independent: When coding; Print all letters as capitals; The letter "aye" is "I"; the number "one" is "1"; The letter "oh" is "O"; the number "zero" is "0". The letter "zee" is " "; the number "two" is "2". Distinguish the letters "U" and "V" carefully. Distinguish the letter S

and the number 5 carefully;

Limit each line of coding in either PL/I or Assembler to columns 1-72. Of course, PL/I conventions require the first coding character to appear in column 2. Use columns 73-80 of each line to contain either the line number or the modification date. This data is automatically entered by a utility program, which is defined later in this section, and should not be entered by the coder.

Use comments liberally throughout all modules. Each module in the system shall have proper identification. The information which should be designated as part of the identification is:

a. TITLE - includes both a six-character module name and a one-line module identifier.

b. COMPANY - is "NEOTERICS, INC. CLEVELAND, OHIO."

c. AUTHOR - identifies the name of the programmer(s). If necessary, use initials.

d. CLIENT - is "NASA LEWIS RESEARCH CENTER."

e. SYSTEM - is "NASA AEROSPACE SAFETY INFORMATION SYSTEM (NASIS)."

f. FUNCTION - describe, in a short paragraph, the purpose and general data flow of the module.

3. Source Language Utility Routine

A source language utility program is provided to enable the programmer to maintain a current, sequenced version of his module. In coding, do not enter any line sequence information.

This utility provides a reorganized and resequenced line data set for the source language and maintains the current version number and date. Thus each modification belongs to a particular version.

Once the source module has been formatted into a line data set, modifications and additions to the module are applied directly to the line data set. During the compiling phase of module development, there is no requirement to use the utility.

However, it is to be executed often enough for each module to maintain a reasonable level of currency in the sequencing of line numbers.

4. Language Dependent

An important consideration in coding is tc provide a module which can be readily modified or maintained. Thus, coding must be modular and well documented. Extra time and care expended in the original coding phase can save many hours of time during debugging or maintenance. The conventions enumerated below in separate language-dependent categories exist tc help the programmer produce a well-structured, self-documenting, and easily readable module.

5. ASSEMBLY LANGUAGE

Several conventions concerning the actual format of the coding are to; Begin be followed. all statement and data labels in column 1; Begin operation codes or macro calls in column 10; Begin operands in column 16. (Macros may need an eight-character operation. In this case, operands start in column 19); Begin comments in column 36. Each line of code should be commented;

To easily identify sub-modules or subroutines, certain organization conventions are to be practiced. Begin all sub-modules or subroutines on a new page on the assembly listing; Within long routines, plan the coding so that a section never crosses a page boundary of the assembled listing; Precede each section with a one-line comment describing the function of the section.

Within the structure dictated by the above conventions, utilize the following standards in coding the modules.

All coding must be re-entrant; Use the standard System/360 linkage conventions for all subroutines. These rules can be found in the IBM Assembler Language Programmer's Guide; The module CSECT should be read-only and contain only executable instructions. Retain all data and constants in the PSECT; Within the PSECT, group the various types of non-executable coding by category. (i.e., all file areas together, all work areas together, all equivalences together, etc.); Identify each file area or structure with

a comment card which precedes the area; Invent meaningful labels for both data and instruction statements in terms of the value or function they represent. Do not use nondescript labels, such as people's names or objects; Define literals, either string or arithmetic, used more than once in the context of the instructions via a data statement or an equivalence prior to its use. (e.g., the character "*" should be defined using an equivalence with the label "ASTERISK"). Minimize the use of instruction counter references. Use of this feature of the language is warranted only in cases where the reference point is within one or two instructions of the instruction having the reference; Make all register references symbolic and conform to the labels R0-R15 for registers 0-15; The standard base register for the PSECT is R13; for the CSECT, R11. Assign additional base registers as desired.

Three macros are available, and should be used, to assist in the standardization of the modules.

a. DBSTART - Code this macro at the very beginning of the module. Its format is:

```
Label   DBSTART   (CSECT entry points),
                  (CSECT base registers),
                  (PSECT base registers)
```

where up to thirty-two entry points, six CSECT registers and six PSECT registers can be specified. Separate multiple operands with commas.

b. DBENTRY - Code this macro as the first instruction in the CSECT defined by the entry points indicated in DBSTART. Its format is:

```
Label   DBENTRY   entry point name
```

c. DBEXIT - Place this macro at the end of the CSECT defined by the entry points indicated in DBSTART. Its format is:

```
Label   DBEXIT
```

All assemblies are to contain both an object listing and a cross-reference listing. A symbol listing is not required in the debugging phase but is to be contained in the final version of the module.

6. PL/I

The format of these modules shall conform to the following rules; All coding must be re-entrant; Begin all statement labels in column 2 of a separate line of code. (Column 1 is used for listing carriage control.); Begin the text of each statement (apart from any label) in column 5 of a new line, subject to the indentation rules which are indicated below; Comment each line to the extent possible. If the line is short enough to permit it, start the comment in column 41. If the comment is too long for the available space, continue it on the next succeeding line; Use indentation to emphasize data and statement relationships and to provide coding which is easily read and understood. Format data structures so that each subordinate level is indented four columns from the margin of the next higher level., Start the THEN and ELSE clauses of the IF statement on a new coding line indented four columns from the margin of the associated IF. Use another four-column indentation for nested IF statements.

Procedures and segments within a module should be easily identifiable. Construct a descriptive header for each segment or procedure in the following format:

One procedure name definition card containing a comment of continuous asterisks following the name and an eject page carriage control character;

One blank comment card or a zero in column 1 of the next card;

One or more comment cards describing the segment and, if necessary, any parameters being passed through the routine;

One blank comment card or a zero in column one of the following card;

One comment card containing a continuous string of asterisks;

If the routine is not a procedure, use a segment label instead of a procedure name on the first card of the header. The first line of statement text then immediately succeeds the header. This convention effectively provides a description box

for each routine.

Standardizatien of the organization and content of the coding provides easy reading and understanding of the modules. Observe these coding rules:

a. Declare all variables.

b. Position all DECLARE statements immediately after the PROCEDURE or BEGIN statements of a block (succeeding any special requirements such as a %INCLUDE LISRMAC(DB); statement when using DBPL/I).

c. Code no more than one statement on a coding line.

d. Qualify all references to minor structures and structure elements, at least, by the major structure name.

Field Name or No.	High-Order Byte (BIT)	Length No.	Length BIT	Length Byte	Contents	Comments
1	1	8		x	File name	
2	9 (0)	1	x		Open switch	
3	9 (1)	1	x		Element switch	Set on to start

Table 1.
Example of a Reference Table chart